Lecture 5. Two-dimensional Arrays

BIF 511 – Programming Fundamentals

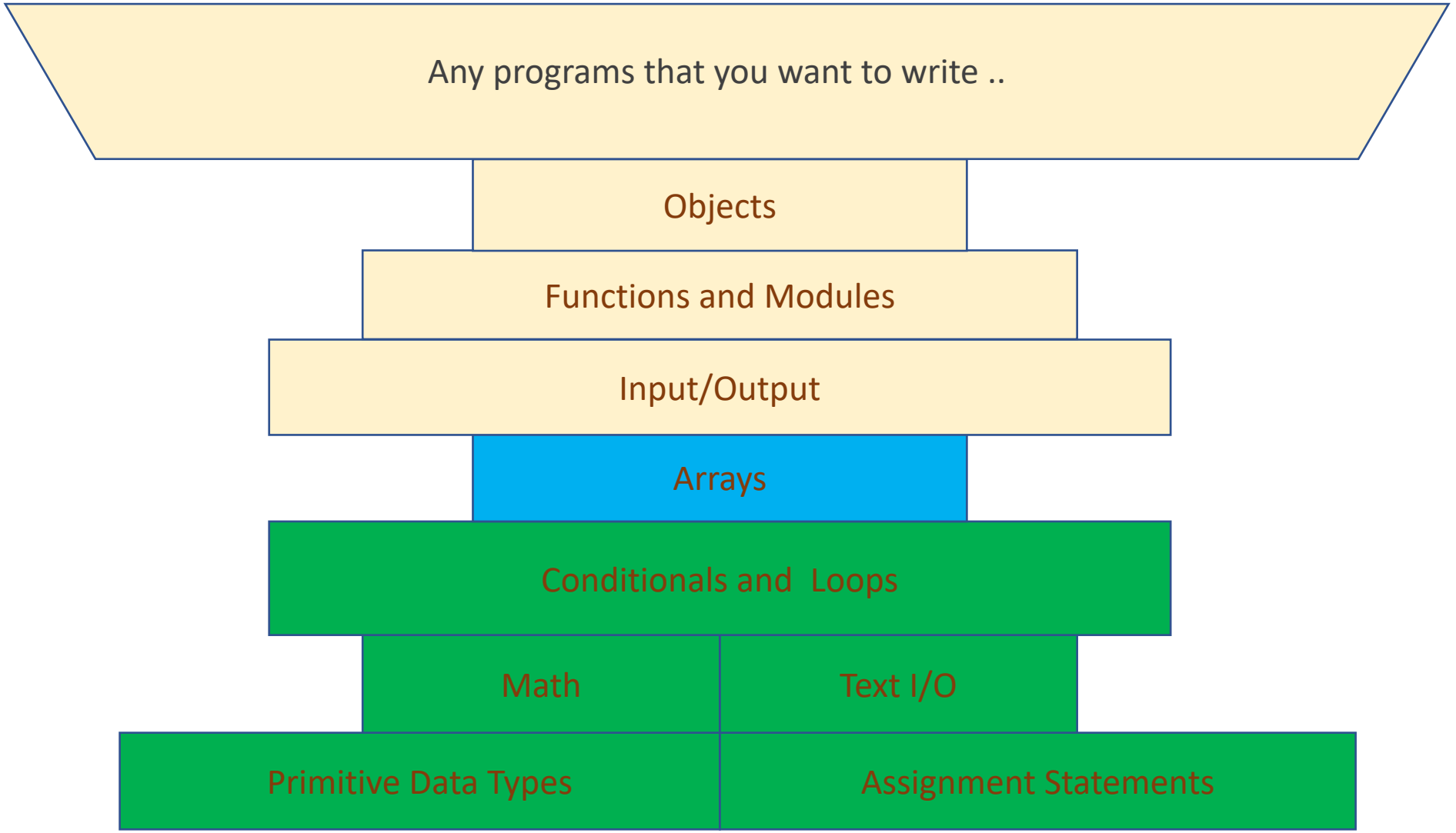
https://sites.google.com/site/kmutt2560bif511/

Kulwadee Somboonviwat

kulwadee.som [at] sit.kmutt.ac.th

** Adapted from computer science lecture slides by Sedgewick R.*

Basic Building Blocks for Programming (in High-level language)

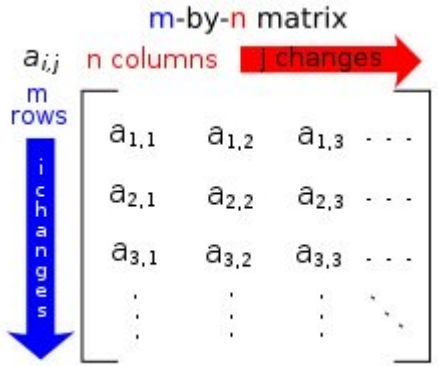


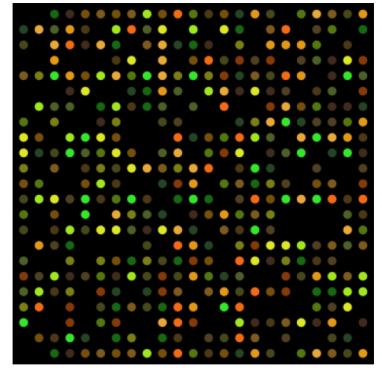
Two-dimensional arrays

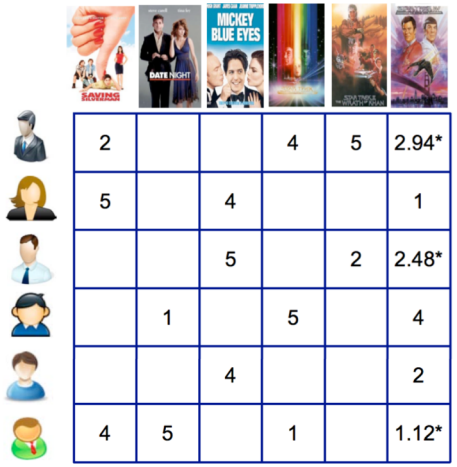
- A two-dimensional array is a doubly-indexed sequence of values of the same type.

Examples.

- Matrices in math calculation
- Outcomes of scientific experiments
- Microarray data
- Pixels in a digital image
- Geographic data
- Transportation network
- Social network relationship
- Item-rating matrices
- ….







Java language support for Two-dimensional arrays

Declare a two-dimensional array	**double[][] a;**
Create a two-dimensional array of a given length	**a = new double[1000][1000];**
Refer to an array entry by index	**a[i][j] = b[i][j] * c[j][k];**
Refer to the number of rows	**a.length;**
Refer to the number of columns	**a[i].length;**
Refer to row *i*	**a[i]**

In Java, we refer to entries of a 2D array with row-major order indexing.

A[0][0]	A[0][1]	A[0][2]	A[0][3]	A[0][4]	A[0][5]	A[0][6]
A[1][0]	A[1][1]	A[1][2]	A[1][3]	A[1][4]	A[1][5]	A[1][6]
A[2][0]	A[2][1]	A[2][2]	A[2][3]	A[2][4]	A[2][5]	A[2][6]

Java language support for Two-dimensional arrays (initialization)

Default initialization to zero for numeric types	**a = new double[1000][1000];**
Default create and initialize in a single statement	**double[][] a = new double[1000][1000];**
Initialize to literal values	**double[][] p =** **{** **{ .92, .02, .02, .02, .02 },** **{ .02, .32, .32, .32, .02 },** **{ .25, .25, .25, .25, .25 },** **{ .10, .20, .20, .10, .40 },** **{ .40, .03, .40, .12, .05 }** **};** **char[][] StudentGrades =** **{ /* row → student, column → subject */** **{'A', 'B', 'B', 'B'},** **{'B', 'C', 'B', 'C'},** **{'A', 'A', 'A', 'A'}** **};**

Application of arrays: (mathematical) vector as Java 1D array

$$\begin{bmatrix} 2 \\ 5 \\ 1 \\ 8 \end{bmatrix} + \begin{bmatrix} 4 \\ 47 \\ 5 \\ 68 \end{bmatrix} = \begin{bmatrix} 2+4 \\ 5+47 \\ 1+5 \\ 8+68 \end{bmatrix} = \begin{bmatrix} 6 \\ 52 \\ 6 \\ 76 \end{bmatrix}$$

```
public class VectorAddition {
  public static void main(String[] args) {
    double[] a = { 2, 5, 1, 8 };
    double[] b = { 4, 47, 5, 68};
    double[] c = new double[a.length];
    for (int i=0; i<c.length; i++)
      c[i] = a[i] + b[i];
    for (int i=0; i<c.length; i++)
      System.out.print(c[i] + " ");
    System.out.println();
  }
}
```

```
> java VectorAddition
6.0 52.0 6.0 76.0
```

Application of arrays: matrix as Java 2D array

$$\begin{bmatrix} 2 & 1 & 3 \\ 3 & 3 & 2 \\ 4 & 1 & 2 \end{bmatrix} + \begin{bmatrix} 7 & 1 & 4 \\ 0 & 5 & 9 \\ 2 & 8 & 7 \end{bmatrix} = \begin{bmatrix} 9 & 2 & 7 \\ 3 & 8 & 11 \\ 6 & 9 & 9 \end{bmatrix}$$

```
> java MatrixAddition
9.0 2.0 7.0
3.0 8.0 11.0
6.0 9.0 9.0
```

```java
public class MatrixAddition {
  public static void main(String[] args) {
    int N = 3;
    double[][] a = {
      {2, 1, 3},
      {3, 3, 2},
      {4, 1, 2}
    };
    double[][] b = {
      {7, 1, 4},
      {0, 5, 9},
      {2, 8, 7}
    };
    double[][] c = new double[N][N];
    for (int i=0; i<N; i++)
      for (int j=0; j<N; j++)
        c[i][j] = a[i][j] + b[i][j];
    for (int i=0; i<N; i++) {
      for (int j=0; j<N; j++)
        System.out.print(c[i][j] + " ");
      System.out.println();
    }
  }
}
```

Application of arrays: (mathematical) vector as Java 1D array

$$\begin{bmatrix} 2 \\ 5 \\ 1 \\ 8 \end{bmatrix} \cdot \begin{bmatrix} 4 \\ 47 \\ 5 \\ 68 \end{bmatrix} = \quad (2x4)+(5x47)+(1x5)+(8x68) = 792$$

```
public class VectorDotProduct {
  public static void main(String[] args) {
    double[] a = { 2, 5, 1, 8 };
    double[] b = { 4, 47, 5, 68};

    double sum = 0.0;
    for (int i=0; i<a.length; i++)
      sum += a[i] * b[i];

    System.out.println(sum);
  }
}
```

```
> java VectorDotProduct
792.0
```

Application of arrays: matrix as Java 2D array

$$\begin{bmatrix} 2 & 1 & 3 \\ 3 & 3 & 2 \\ 4 & 1 & 2 \end{bmatrix} \times \begin{bmatrix} 7 & 1 & 4 \\ 0 & 5 & 9 \\ 2 & 8 & 7 \end{bmatrix} = \begin{bmatrix} 20 & 31 & 38 \\ 25 & 34 & 53 \\ 32 & 25 & 39 \end{bmatrix}$$

```
> java MatrixMultiplication
20.0 31.0 38.0
25.0 34.0 53.0
32.0 25.0 39.0
```

```java
public class MatrixMultiplication {
  public static void main(String[] args) {
    int N = 3;
    double[][] a = {
      {2, 1, 3},
      {3, 3, 2},
      {4, 1, 2}
    };
    double[][] b = {
      {7, 1, 4},
      {0, 5, 9},
      {2, 8, 7}
    };
    double[][] c = new double[N][N];

    for (int i=0; i<N; i++)
      for (int j=0; j<N; j++)
        for (int k=0; k<N; k++)
          c[i][j] += a[i][k]*b[k][j];

    for (int i=0; i<N; i++) {
      for (int j=0; j<N; j++)
        System.out.print(c[i][j] + " ");
      System.out.println();
    }

  }
}
```

Pop-Quiz: How many multiplications to multiply two N-by-N matrices ?

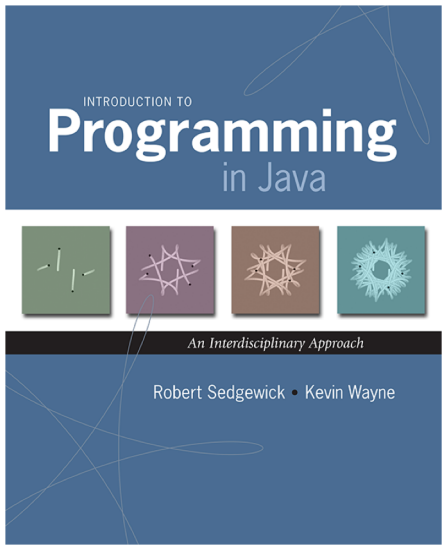
```
double[][] c = new double[N][N];

for (int i=0; i<N; i++)
  for (int j=0; j<N; j++)
    for (int k=0; k<N; k++)
      c[i][j] += a[i][k]*b[k][j];
```

1. N
2. N^2
3. N^3
4. N^4

Summary: Arrays

- Arrays: a basic building block in programming
  - Enable storage of large amounts of data (values all of the same type).
  - Efficient access to an array element with an index
- Applications in science and engineering
  - Matrix computation
  - Image processing
  - Microarray data analysis
  - ….

References



- Sedgewick and Wayne. Introduction to Programming in Java – an Interdisciplinary Approach

Image Credits

https://www.decodedscience.org/wp-content/uploads/2013/12/xA04_matrix_jpeg.jpg.pagespeed.ic.8wTjsg2sbC.jpg

https://ffp4g1ylyit3jdyti1hqcvtb-wpengine.netdna-ssl.com/ux/files/2014/04/recommendationExample-1.png